ARMY RESEARCH LABORATORY

Dismounted Infantry Visualization Research: The Dismounted Infantry Simulation (DISim)

by Mark A. Thomas

ARL-TN-193 December 2002

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory

Aberdeen Proving Ground, MD 21005-5067

ARL-TN-193 **December 2002**

Dismounted Infantry Visualization Research: The Dismounted Infantry Simulation (DISim)

Mark A. Thomas

Computational and Information Sciences Directorate, ARL

Contents

List of Figures

List of Tables

1. Introduction

The U.S. Army Research Laboratory (ARL) is conducting research and development of simulation and visualization tools for the training and mission rehearsal of Dismounted Soldiers in virtual, distributed simulation. The focus of this research is on the development of general-purpose algorithms to produce realism for the soldier in battlefield situations to include open terrain and urban environments. The algorithms developed to date include dynamic terrain effects of cratering and wall breaches, shadows, sound effects, and exertion. The software tool used to develop, test, and benchmark these algorithms is the Dismounted Infantry Simulation Test-Bed (DISim).

The DISim is software. The software is engineered to allow the easy integration of new algorithms for testing and evaluation. The DISim is also a battlefield visualization system providing entity icons and behaviors, munitions effects, sound effects, movement, and environmental effects. The DISim has interfaces to distributed simulation network protocols such as distributed interactive simulation (DIS) and the high-level architecture (HLA). The DISim has interfaces to external virtual reality equipment such as helmet mounted displays (HMD), position trackers, and weapons triggers.

This document will give an overview of the DISim battlefield visualization capabilities, software architecture, algorithms developed using it, and run-time considerations.

2. Battlefield Visualization Capabilities

The DISim is a three-dimensional (3-D) virtual battlefield visualization with the following features:

- Terrain visualization,
- Vegetation, roads, and rivers,
- Urban environments including multistory buildings, sewers, roads, and other built-up environmental objects,
- Munitions effects of blast, smoke, and sound,
- Environmental effects of background noise, fog, clouds, time-of-day, and shading,
- Distributed simulation entities. The current implemented entities are friendly and enemy dismounted combatants, civilians, M1A1 Abrams tanks, M2 Bradley Fighting Vehicles, AH-64 Apache Helicopters, T80/T72 tanks, and others as needed,

- Dynamic terrain effects of cratering, dings and nicks, breaches, and markings,
- Scrolling text boxes for kiosks and signs,
- Wildlife including deer, ground hogs, and flocks of birds,
- HMD and motion tracking for true immersion of the viewer,
- Keyboard and mouse personal computer (PC) interactive controls,
- Inset view of the soldier's posture,
- Heads-up display showing position, orientation, and weapons status,
- 3-D sound from the viewer's position and orientation,
- Weapons fire modeling to include tracers, shell casings, and impact indications, and
- Mobility using keyboard, mouse, or external devices such as the omni-directional treadmill.

This list grows as new capabilities are added. Figure 1 shows a view of the scene in DISim of the Ft. Polk Shugartt-Gordon database.

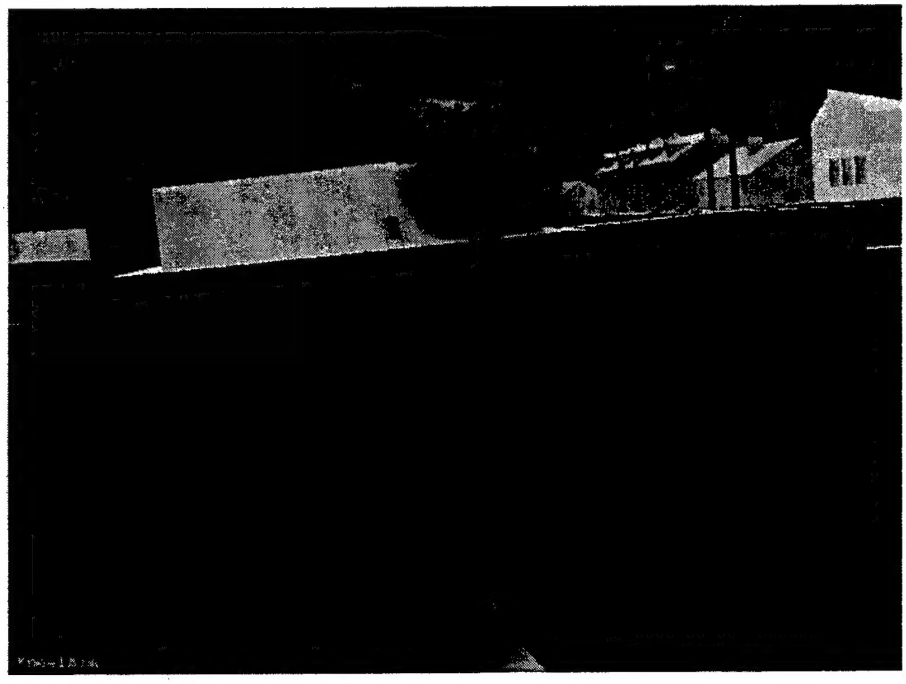

Figure 1. DISim visualization of Ft. Polk database.

3. Software Architecture

The DISim is developed through the ground observer's viewpoint. The software is tuned for the head and body movements of a ground observer. With this constraint in mind, the software requires collision detection with the ground and objects, a viewpoint having the head/neck constraints of an individual in various postures, and the ability to climb, jump, or crawl around or through obstacles and passages. The sounds the viewer hears while walking, running, standing still, firing a weapon, or sounds generated by other entities on the battlefield are important. With these considerations in mind, the central data structures in DISim are centered around the viewer and his weapon.

DISim is written in gnu C++ on the RedHat Distribution of the Linux operating system and is portable. The software was developed using classes and the standard template library (STL) for data structures, methods, queues, and lists. The test-bed uses industry standard software engineering to provide robust operation, integration of components, and maintainability.

3.1 Computer Graphics

Visualization is handled using the Silicon Graphics Incorporated (SGI) Performer programming interface visualization libraries [1]. Performer handles graphics display generation, graphics model loading, collision detection, keyboard and mouse input/output. Performer includes high-performance functions to control lighting, shading, texturing, and 3-D perspective views.

Built on the SGI OpenGL graphics standard, Performer allows the use of OpenGL calls in the draw process. This allows the application designer to implement heads-up displays and user-developed draw functions.

DISim divides graphics into four main components: terrain, terrain objects, environmental entities, and dynamic entities. Terrain is the ground on which all structures stand. DISim uses terrain for computing height above terrain (ground clamping). Terrain objects are independent objects the user can interact with. These include trees, vegetation, buildings, and other structures. Environmental entities are synthetic natural environment icons intended to aid immersion in the simulation such as clouds, kiosks, fog, and rain. Dynamic objects have dynamic state, such as position, velocity, and orientation. These states are updated periodically to reflect changes in position, condition, or other factors relevant to the entity such as friendly and enemy dismounted combatants, ground platforms, and air vehicles. DISim processes the different components as follows (Table 1):

Table 1. DISim graphics objects processing.

Component	Entity Collision	Dynamic Effects	Dead Reckoning
Terrain	Yes	No	No
Environmental	No	No	Yes
Terrain Objects	Yes	Yes	No
Dynamic Objects	Yes	Yes	Yes

Table 1 shows the processing of the various type of graphics objects. Entity collisions stop the viewer if he comes into contact with an object. Dynamic effects are breaches, cratering, and rubble creation. Dead reckoning is state integration.

From a system performance perspective, entity collision is the heaviest processing load. Collision detection in dynamic environments is an expensive operation to perform. So the addition of environmental entities to the simulation is an effective way of adding objects to the simulated world without adding processing latency to the system.

3.2 Human Figure Animation

Character animation uses the Boston Dynamic DI-Guy software libraries [2]. DI-Guy provides realistic characters (soldiers, as well as male and female civilians) that conform to the standard enumerations in DIS [3]. The characters walk, run, stand, kneel, lie prone, and fire weapons using realistic animations. The head and weapon can be aimed using the DI-Guy software application-programming interface (API), and the character animations transfer smoothly between postures and actions. The DISim gets head and neck azimuth and elevation directly from the DI-Guy soldier using the diguy _ get _ link _ position() function.

3.3 System Flowchart

The remainder of the software is in-house ARL software developed by ARL researchers. The software uses classical animation software flow of system initialization, then main-loop. Figure 2 shows the flowchart of DISim.

3.4 Networking

Networking in DISim uses a second program, the Watcher. The Watcher can use DIS and/or HLA protocol to send and receive entity state. The DISim uses shared memory to transfer information to and from the Watcher. The shared memory messages are used to add and delete entities from the simulation, update entity information, send and receive weapons fire and

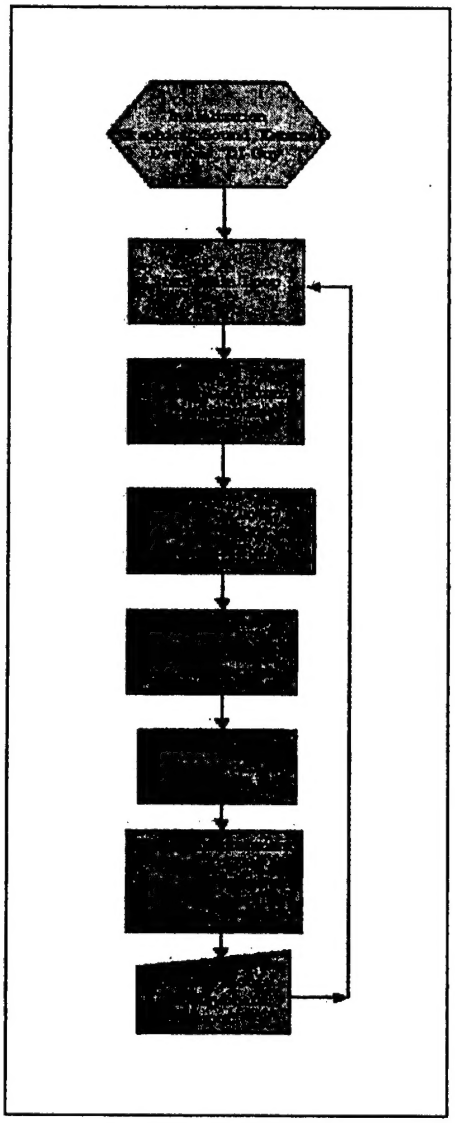

Figure 2. DISim flowchart.

detonation messages, and receive dynamic terrain polygon information. The use of the Watcher allows DISim to be run stand-alone or networked without modification or set-up. The shared memory interface provides a tight coupling between the programs and shields the visualization system from unnecessary or erroneous data received over the network. For example, the Watcher loads the DISim models definition file. When the Watcher receives an ENTITY STATE Protocol Data Unit or HLA discover object message, it checks the database of models for DISim. If DISim does not have a model to represent the entity, the Watcher prints a warning message to the console and ignores data updates from the entity.

The use of the Watcher also provides a safety valve for data dropouts or network downtime. The user in the DISim can continue to maneuver and fight while the network is being reconnected.

The Watcher also handles coordinate conversion from local to world coordinates, maintains a database of entities discovered in the simulation, and transmits DISim location, position, appearance, and weapons firings to the network.

4. Current Algorithms and Libraries

DISim is a software test-bed. The author feels that proper benchmarking is performed under real-world conditions. This means for an algorithm to be acceptable for the purpose of dismounted warrior simulation, it must execute without adversely affecting the runtime of DISim. The computation of breaches, rubble generation and distribution, cratering, dynamic objects, sound effects, and distributed simulation entity updates must execute within the 30–60 frames/s update rate of the simulation.

DISim contains and uses ARL-developed and modified algorithms to compute and display wall breaches, rubble and debris, provide mobility over the terrain for dismounted maneuvers, terrain cratering, and visual objects. The creation of a sensory-rich virtual environment for the ground warfighter requires high resolution for realism.

4.1 Breaching and Rubble

The insertion of breaches uses the ARL Hole Library and rubble distribution methodology [4]. These general purpose algorithms compute and format holes in polygons, and the removed material is used to generate the resultant volume of rubble. The DISim software formats the input to the library routines, executes them, then integrates the results into the new graphics database display list. The DISim was used to develop and test these algorithms so to ensure their execution did not adversely affect the runtime of the visualization system. To test the algorithms, the researcher maneuvers to a wall, selects the AT8 weapon, and fires at the wall. The DISim formats a fire interaction message and processes it. When the AT8 round hits the wall, DISim formats a detonation interaction message and processes it. The munition type and velocity vector are parsed from the detonation interaction message. The polygon belonging to the wall is found by using the collision detection routines in Performer. These data are sent to the hole library for processing. When the hole generation is completed, the impacted polygon is deleted from the display list, and the new wall polygons are inserted. The rubble generated from the breach is added to the display list for visualization.

4.2 Keyboard/Mouse Mobility

Mobility over the terrain employs a number of techniques. The dismounted combatant can walk, run, crawl, climb, and jump. The DISim enforces proper locomotion behavior using ground

clamping, collision detection, and empirical methods. The ground speed uses the values from the DI-Guy software. These speeds are as follows:

- 0.19 m/s crawl,
- 0.385–1.27 m/s walk,
- 1.3–2.0 m/s jog, and
- 2.0 + run.

The viewer selects these speeds using the "r" key on the keyboard. The icon and viewpoint are updated based on the speed over ground. For instance, if the viewer is lying down, then gets up to run, as the speed accelerates from 0.0 m/s to 2.0 m/s, the icon crawls, then stands to walk, then transitions to a jog motion.

The viewer actor is prevented from walking through objects. This is accomplished using three collision vectors. The collision vectors are at the head, waist, and ankles. These locations provide a three-point check on forward motion. If two intersections are detected, the actor stops. The head vector follows the height of the head. This allows an actor to crawl through holes. The vector at the ankles forces the viewer to jump over ledges, and the vector at the waist prevents the actor from walking through railings. The use of collision vectors at the head, waist, and ankles provide successful collision detection in the Ft. Polk and Ft. Benning McKenna military operations on urban terrain (MOUT) site databases.

A major consideration in locomotion is climbing and descending ladders and ropes. DISim provides this capability using keyboard commands. To climb an object, the viewer must be in contact with the surface of the object. This is accomplished by simply walking up to it. The F9 key initiates the climb. When the actor reaches the top of the object, he steps onto it by pressing the left mouse while simultaneously hitting the "j" (jump) key.

4.3 External Device Control

Mobility may be controlled by external devices such as the omni-directional treadmill [5]. In this mode, the keyboard and mouse mobility controls are disabled. The DISim reads position and speed information from shared memory. It is the responsibility of the external device driver to update the shared memory with appropriate data.

This mode is most useful when the scene is viewed through an HMD. The HMD with a tracker provides the necessary data for viewpoint positioning. The DISim is compatible with any HMD that can handle super video graphics adapter (SVGA) mode.

DISim reads the data from shared memory and updates the viewpoint. External data includes the following:

- velocity X and Y,

- head X, Y, and Z position,
- head yaw, pitch, and roll,
- gun X, Y, and Z position,
- gun yaw, pitch, and roll,
- gun trigger status,
- posture, and
- weapon selection.

Based on the information contained in shared memory, the user can walk, run, look around using an HMD, and fire at targets. The shared memory can also accommodate posture commands if available.

4.4 Dynamic Objects

Another focus of research is dynamic objects. A dynamic object is simply something in the environment providing sensory stimulation for the participant. When outside in the woods, one would see and hear wildlife walking, birds flying, and trees blowing in the wind. In an urban setting, billboards, kiosks, and scrolling text signs are prominent. These objects are included in video games to enhance the realism factor. DISim does the same for military simulation. Dynamic objects in DISim include wildlife, moving skyscapes, various rotating objects, and kiosks with scrolling text. These objects are developed in classes to facilitate ease of modification and extension. The dynamic objects are encapsulated in standard template library lists, providing a standard method for accessing and updating them.

4.5 3-D Sound

Three-dimensional sound is important. The use of the sound modality in battlefield visualization is an active research topic. ARL is investigating the use of the OpenAL sound library developed by Creative Labs and Loki Entertainment Software to create an open source sound library for use in game programming [6]. The OpenAL uses the modeling semantics of OpenGL (e.g., using modeling transformations to define coordinate axes about which sounds are generated and directional effects computed). The OpenGL semantics of OpenAL provide a seamless integration with DISim because SGI Performer is a library built on OpenGL. This means coordinates and directions in DISim are directly translated to the OpenAL sound library for processing. The use of OpenAL provides the user of DISim with 3-D sound. The user can stand in position while sound travels around him, hear firing and detonation events from the proper location, and distinguish sounds close or far away.

The DISim currently has the following sounds implemented:

- background sounds (e.g., forest, ocean),
- weapons fire (e.g., small arms, machine gun, artillery),
- vehicle sounds (e.g., tanks, helicopters), and
- dismounted sounds (e.g., footsteps, collision).

5. Run-Time Considerations

The DISim requires a PC-running RedHat Linux 7.1+, SGI Performer for Linux software and license, and the DIGuy software and license.

5.1 Initialization Files

The software requires many files to run. These files are terrain database, player initial values, and program configuration. The use of a shell script is recommended. Figure 3 presents the script to run DISim using the Ft. Polk Shugartt-Gordon database.

```
#!/bin/csh
setenv MHOME /home/disim/models
setenv DMODELS $MHOME/AH64:$MHOME/BLACKHAWK:$MHOME/BMP-
1:$MHOME/BRADLEY:$MHOME/BRDM-
1:$MHOME/ENV:$MHOME/F18:$MHOME/F22:$MHOME/FLT:$MHOME/HI
ND:$MHOME/HMMWV:$MHOME/M1ABRAMS:$MHOME/MIG27:$MHOM
E/RUBBLE:$MHOME/SOLDIER:$MHOME/T_72:$MHOME/T_80:$MHOME/
RUBBLE:$MHOME/cars
setenv TERRAIN /home/disim/FTPOLK: \
/home/disim/FTPOLK/models:$DMODELS
setenv PF_PATH /home/disim/Viewer/standard: \
/home/disim/texture:/usr/share/Performer/data:$TERRAIN
setenv BDI /usr/local/bdi
../../bin/viewer -t /home/disim/FTPOLK/models/fortpolk32vs.flt -m models.dat -p player.dat -v 0
-s /home/disim/Viewer/scenario/FtPolk/sound.dat
```

Figure 3. Shell script example.

This script sets the appropriate environmental variables for the proper execution of the program. The PF_PATH environment variable is the file search path for the Performer libraries. All graphics models to be used in the simulation must be included in PF_PATH.

The BDI environmental variable points to the DI-Guy home directory. The command line options to DISim are as follows:

- -t filename – This is the filename for the terrain. Up to fifty files may be loaded using multiple -t entries. Terrain files may be OpenFlight (.flt), Performer File Binary (.pfb), or any other file format currently supported by SGI Performer.
- -m modelsFilename – This file contains model definitions. It includes entries for each model to be represented in the simulation, both internal and network entities. Figure 4 is a sample model file entry.
- -p playerFilename – This file contains identification information for the player, the weapons he carries, and initial location. Figure 5 is a player file for the Ft. Polk Shugartt-Gordon database.
- -v 0 or 1 – This option controls mobility. A value of 0 enables keyboard/mouse mobility (the default), and a value of 1 enables external device mobility.
- -s soundFileName – This option identifies the sound configuration file (Figure 6).

There is one other initialization file. It is the environmental entities file.

5.2 Environmental Entities File

This file is not listed on the command line, but must exist in the directory where the run script is started. The filename is environ.dat. This file contains the list of environmental entities to add to the simulation. Environmental entities are used for adding rotating cloudscapes, night skyscapes, flying birds, and animals. Other types of environmental entities are signs with scrolling texts. These entities are extremely lightweight, yet add realism to the synthetic environment.

The five classes of environmental entity are weather, time-of-day, bird, critter, and text. A weather environmental entity is characterized by rotation. This is used for rotating cloudscapes, rotating billboards, or any other entity that has a fixed location and rotates about its base. A time-of-day entity may be turned on or off depending on time of day. This is useful for turning a night sky or streetlights on or off. A bird entity is a rotating entity at a fixed altitude. A critter is a rotating entity that clamps to the ground. A text entity is an object that displays text from a file in either a horizontal or vertical scroll at a specified rate.

Rotation for weather entities is about its base with zero radius. Bird and critter entities rotate around a point at a specified radius. The period of revolution for all rotating entities is user settable in the initialization file. Figure 7 is a sample environ.dat file.

```
START_ENTITY
TYPE M1_Abrams
MODEL m1-high.flt
CLASS tank
POSITION 0.0 0.0 0.0
ORIENTATION 0.0 0.0 0.0
DIS_ENUMERATION 1 1 225 1 1 0 0
SCALE 1.0
END

START_ENTITY
TYPE T-80
MODEL T-80.flt
CLASS tank
POSITION 0.0 0.0 0.0
ORIENTATION 0.0 0.0 0.0
SCALE 1.0
DIS_ENUMERATION 1 1 222 1 1 0 0
END

START_ENTITY
TYPE T-72
MODEL T-72.flt
CLASS tank
POSITION 0.0 0.0 0.0
ORIENTATION 0.0 0.0 0.0
SCALE 1.0
DIS_ENUMERATION 1 1 222 1 2 0 0
END
```

Figure 4. Models definition file.

5.3 Controls

The software functions like a video game, with keyboard and mouse control. The keyboard commands to DISim are as follows:

- a/A-Accelerate,
- e-Advance time of day,

```
name DIVirtura
hot dynamic
location 502.0 500.0 0.0
dis_enumeration 3 1 225 1 32 1 0 0
weapon M16A2 2 8 225 2 1 5 0 11 20 10 3.0
weapon AT8 2 2 225 2 8 1 0 5 3 3 0.2
weapon LOSAT 2 2 225 1 10 0 0 0 10 3 1.0
weapon MARKER 2 10 225 2 8 1 0 5 1 100 1.0
```

Figure 5. Sample player file.

```
// Sound config file for DISim
SoundsOn Yes
// This directory is prepended to all sound files
// This line must be before all other locations in this file
SoundDir /home/disim/Viewer/sounds

MunitionsFile /home/disim/Viewer/standard/munitions.dat

AmbientSoundOn No
AmbientSound ambient/birds.wav

MusicOn Yes
Music ambient/JUNGLE.WAV

DefaultSound rifle.aiff.wav
CollisionSound uhh.aiff.wav
FootstepsSound footsteps.wav
```

Figure 6. Sample sound configuration file.

- f-Advance fog,
- F-Toggle fog,
- g-Toggle collision on/off,
- h-Print commands,
- j-Jump,
- l-Toggle light,

```
START_ENTITY
ID 1
NAME NIGHT_SKY
LOCATION 400.0 400.0 0.0
HEADING 0.0 0.0 0.0
SPEED 1.0
SCALE 1.5
START_POSITION 400.0 400.0 0.0
MODEL /home/models/FLT/nightsky.flt
TYPE TOD
END

START_ENTITY
NAME CLOUDS
LOCATION 400.0 400.0 0.0
HEADING_RATE 2.0 0.0 0.0
SCALE 1.0
SPEED 0.0
START_POSITION 400.0 400.0 0.0
MODEL /home/models/FLT/sky2.flt
TYPE WEATHER
END

START_ENTITY
ID 1
NAME Weather
LOCATION 357.0 550.0 30.0
HEADING 0.0 0.0 0.0
HEADING_RATE 10.0 0.0 0.0
SPEED 0.0
SCALE 1.0
START_POSITION 357.0 550.0 30.0
MODEL /home/models/FLT/arl_sign.flt
TYPE WEATHER
END

START_ENTITY
ID 1
NAME KIOSK
LOCATION 510.1 524.9 14.0
HEADING 29.0 0.0 0.0
SPEED 1.0
SCALE 0.5
RATE 1.0
SCROLL_METHOD 0
START_POSITION 509.7 524.9 14.0
MODEL /home/models/FLT/arl_sign.flt
TYPE TEXT
FILENAME ads.dat
END
```

Figure 7. Sample environ.dat file.

- L-Print entity list,
- s-Stop,
- G-Show graphics statistics,
- M/D-Mount/dismount vehicle,
- r-Advance speed,
- c-Release window focus,
- C-Set position to the center of the database,
- R-Request all DIS entities,
- \-Select weapon,
- <space bar>-Fire weapon,
- F1-Set posture: posture stand,
- F2-Set posture: posture kneel,
- F3-Set posture: posture prone,
- F5-Set posture: posture standaim,
- F6-Set posture: posture kneelaim,
- F7-Set posture: posture proneaim,
- F9-Climb. Must be stopped at an object,
- F10-Descend. Must be on top of an object,
- F11-Raise view 10 m,
- F12-Drop view 10 m,
- Left arrow-Strafe left, and
- Right arrow-Strafe right,

The mouse commands are as follows:

- Left button-Move forward,
- Right button-Move backward,
- Mouse up-Raise head, and
- Mouse down-Lower head.

5.4 Weapons Fire

In all experiments and exercises with live infantry soldiers, their main issue is how to aim and fire the weapon. In DISim, the keyboard space bar fires a round. The backslash (\) key selects a weapon. The mouse controls azimuth and elevation of the weapon. DISim puts a red crosshair on the bullet impact location, acting as a laser sight. The user simply aims the crosshair at a target then presses the space bar. The cross-hair sight can also be controlled by an external mock-up rifle.

6. Conclusions

The DISim software test-bed has been used to develop algorithms to aid in the realism and training effectiveness of dismounted combatants. The software was used at the Simulation and Modeling for Acquisition, Requirements and Training 2001 Research and Development Engineering Center Federation technology demonstration in April 2001 [7] and in soldier evaluations at the Ft. Benning Land Warrior Test-Bed in September 2001. The DISim will continue to be used to develop and test algorithms to include vegetation, rain and snow, mobility impediments, improved rubble and debris, and lighting and illumination effects.

DISim is serious software for a serious purpose. Its purpose is to provide the scientific and computing foundation for algorithm development to enable our warfighters with the best chance of survivability and mission success.

This report is an overview of the DISim software, capabilities, and execution instructions. It is not exhaustive, however, as all research tools change over time.

7. References

1. Clay, S. et al. "IRIS Performer 2.0 C++ Reference Pages." Document No. 007-2782-001, Silicon Graphics Incorporated, Mountain View, CA, 1995.
2. Boston Dynamics, Inc. "DI-Guy 4 User Manual." Boston Dynamics, Inc., 614 Massachusetts Avenue, Cambridge, MA, 2000.
3. Institute for Simulation and Training. "Enumeration and Bit-Encoded Values for Use With IEEE1278.1-1994, Distributed Interactive Simulation – Application Protocols." IST-CR-93-46, Institute for Simulation and Training, Orlando, FL, 1993.

4. Neiderer, A. M., and C. Hansen. "Distribution of Fragments Resulting From Polygonal Object Fracture." ARL-TN-182, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, September 2001.

5. Darken, R. P., W. P. Cockayne, and D. Carmein. "The Omni-Directional Treadmill: A Locomotion Device for Virtual Worlds." *Proceedings of the Association for Computing Machinery User Interface Software and Technology*, UIST 1997, pp. 213–221, 1997.

6. Grantham, J. "OpenAL Home Page." <http://www.openal.org/about, 2000>.

7. Sauerborn, G. "Modifications of the Lethality Server for Initial RDEC Federation Integration." ARL-MR-522, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, Appendix A, p. 30, December 2001.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 2002	Final, 10 January 1998–1 April 2002

4. TITLE AND SUBTITLE
Dismounted Infantry Visualization Research: The Dismounted Infantry Simulation (DISim)

5. FUNDING NUMBERS
JONO 3TEDNC

6. AUTHOR(S)
Mark A. Thomas

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRL-CI-CT
Aberdeen Proving Ground, MD 21005-5067

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TN-193

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)

10.SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT*(Maximum 200 words)*

This report is an overview of the Dismounted Infantry Simulation (DISim) Software Test-Bed. DISim is software that allows researchers to test new simulation and visualization algorithms for real-time virtual simulation. Targeted to the simulation needs of the ground warfighter, the algorithms developed using the DISim test-bed are high-resolution real-time effects for simulation, training, and mission rehearsal. C++ based on the Linux operating system, the software uses commercial software for graphics and realistic human figure animation. This report will introduce the DISim, detail its battlefield simulation features, describe the software architecture, and present a brief overview of algorithms developed using it.

14. SUBJECT TERMS
visualization, dismounted infantry, modeling and simulation, DIS, high-level architecture

15. NUMBER OF PAGES
22

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL